WE CLAIM:

1. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks for each said program, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a plurality of contexts (660), each of said contexts being assigned to one of said plurality of programs for processing one of said programs, each of said contexts having at least a plurality of registers and a plurality of condition code storages for containing processing status information,

a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the program instruction stream of said assigned program from said adding means, each of said logical resource drivers comprising:

(a) a plurality of queues (1560), and

(b) means (630, 620) operative on said plurality of said basic blocks containing said intelligence from said adding means for delivering said instructions in each said basic block into said plurality of queues based on said logical processor number, said instructions in each said queue being entered according to said instruction firing time wherein the earliest instruction firing time is entered first,

a plurality of individual processor elements (PEs), each of said processor elements being free of any context information,

means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions with the earliest instruction firing time first in said queues from each of said logical resource drivers, in a predetermined order, to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

first means (670) for connecting each of said processor elements with any one of said plurality of contexts, each of said processor elements being capable of accessing said plurality of registers and condition code storages in a program's context during the processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

2. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks for each of said programs, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a plurality of contexts (660), each of said contexts being assigned to one of said plurality of programs for processing one of said programs, each of said contexts having at least a plurality of registers and a plurality of condition code storages for containing processing status information,

a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the program instruction stream of said assigned program from said adding means, each of said logical resource drivers comprising:

(a) a plurality of queues (1560), and

(b) means (630, 620) operative on said plurality of said basic blocks containing said intelligence from said adding means for delivering said instructions in each said basic block into said plurality of queues based on said logical processor number, said instructions in each said queue being entered according to said instruction firing time wherein the earliest instruction firing time is entered first,

a plurality of context free individual processor elements (PEs),

means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions from each of said logical resource drivers, in a predetermined order, to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

first means (670) for connecting each of said processor elements with any one of said plurality of contexts, each of said processor elements being capable of accessing said plurality of registers and condition code storages in a program's context during said processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

3. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks for each of said programs, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a plurality of contexts (660), each of said contexts being assigned to one of said plurality of programs for processing one of said programs, each of said contexts having a plurality of registers and a plurality of condition code storages for containing processing status information,

a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the program instruction stream of said assigned program from said adding means for storing said instructions, and fetching, in order of said instruction firing time, said instructions in each basic block, and delivering said instructions according to the logical processor number for each instruction,

a plurality of individual context free processor elements (PEs), means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions from each of said logical resource drivers, in a predetermined order, to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

first means (670) for connecting each of said processor elements with any one of said plurality of contexts, each of said processor elements being capable of accessing said plurality of registers and condition code storages in a program's context during said processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

4. The parallel processor system according to Claims 1, 2, or 3 in which:

said adding means is further capable of statically adding shared context storage mapping (S-SCSM) information to each said instruction, said statically shared context storage information containing level information for each said instruction in order to identify the different program levels contained within each said program, said contexts having a different set of registers for each said program level,

each said set of registers being identified by said statically added shared context storage mapping information, and

said processor elements being further capable of processing each of its instructions in a set of registers identified by said statically added shared context storage mapping information.

5. The parallel processor system according to Claims 1, 2, or 3 in which:

each of said logical resource drivers further comprises means for dynamically adding shared context storage mapping (D-SCSM) information to each said instruction, said dynamically added shared context storage information containing the identity of the context assigned to the programs contained within each said logical resource driver,

each of said contexts being assigned to one of said logical resource drivers, each said context being identified by said dynamically added shared context storage mapping information, and

said processor elements being further capable of processing each of its instructions in the context identified by said dynamically added shared context storage mapping information.

6. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for adding intelligence to each instruction in each of said plurality of basic blocks, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a plurality of contexts (660), each of said contexts being assigned to one of said plurality of programs, each of said contexts having a plurality of register resources,

a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the instruction stream of said assigned program from said adding means for storing said instructions in each basic block according to the logical processor number for each instruction,

a plurality of individual context free processor elements (PEs),

means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions from each of said logical resource drivers, in a predetermined order, to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

first means (670) for connecting each of said processor elements with any one of said plurality of contexts, each said processor elements being capable of accessing said resources in a program's context during said processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

7. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for adding intelligence to each instruction in each of said plurality of basic blocks,

a plurality of contexts (660), each of said contexts being assigned to at least one of said plurality of programs, each of said contexts having a plurality of register resources, a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the program instruction stream of said at least one assigned program from said adding means for storing said instructions in each basic block,

a plurality of individual processor elements (PEs),

means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions from each of said logical resource drivers, to individually assigned processor elements,

first means (670) for connecting each of said processor elements to any one of said plurality of contexts, each said processor elements being capable of accessing said resource in a program's context during said processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

8. The parallel processor system according to Claims 6 or 8 in which:

said adding means is further capable of adding information to each said instruction, said information containing level information for each said instruction in order to identify the different program levels contained within each said program,

said contexts having a different set of register resources for each said program level,

each said set of resources being identified by said added information, and

said processor elements being further capable of processing each of its instructions in a set of register resources identified by said added added information.

9. The parallel processor system according to Claims 6 or 8 in which:

each of said logical resource drivers further comprises means for adding information to each said instruction, said added information containing the identity of the context assigned to the programs contained within each said logical resource driver,

each of said contexts being assigned to one of said logical resource drivers, each said context being identified by said added information, and

said processor elements being further capable of processing each of its instructions in the context identified by said added information.

10. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a plurality of programs in said system, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for adding intelligence to each instruction in each of said plurality of basic blocks, said added intelligence containing level information for each said instruction in order to identify the different program levels contained within each said program,

a plurality of contexts (660), each of said contexts being assigned at least one of said plurality of programs, each of said contexts having a plurality of register resources, with a different set of register resources for each said program level, each said set of resources being identified by said added intelligence,

a plurality of logical resource drivers (LRDs) with each logical resource driver being assigned to one of said plurality of contexts, each of said logical resource drivers being receptive of said basic blocks corresponding to the program instruction stream of said at least one assigned program from said adding means for storing said instructions in each basic block, each of said logical resource drivers being further capable of adding information to each said instruction, said added information containing the identity of the context assigned to the programs contained within each said logical resource driver,

a plurality of individual processor element (PEs),

means (650) connecting said plurality of processor elements to said plurality of logical resource drivers for transferring said instructions from each of said logical resource drivers to individually assigned processor elements,

first means (670) for connecting each of said processor elements to any one of said plurality of contexts, each said processor elements being capable of accessing the set of resources, identified by said added intelligence, in a program's context, identified by said added information, during said processing of the program's instruction,

a plurality of memory locations (610), and

second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable of accessing said memory locations during said processing of each said instruction.

11. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a program, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a logical resource driver (LRD) receptive of said basic blocks corresponding to the program instruction stream from said adding means for storing said instructions and said logical resource driver being further capable of fetching, in order of said instruction firing time, said instructions in each basic block, and delivering said instructions according to the logical processor number for each instruction,

a plurality of individual context free processor elements (PEs),

means (650) connecting said plurality of processor elements with said logical resource driver for transferring said instructions from said logical resource driver to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

a plurality of shared storage resources (660),

first means (670) for connecting each of said processor elements with any one of said plurality of resources, each said processor elements being capable of accessing any one of said resources during said instruction processing,

a plurality of memory locations (610), and second means (620, 630) for connecting each of said processor elements with any one of said plurality of memory locations, each said processor element being capable or accessing said memory locations during said instruction processing.

12. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a program, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks, said added intelligence at least having a logical processor number (LPN) and an instruction firing time (IFT),

a logical resource driver (LRD) receptive of said basic blocks corresponding to the program instruction stream from said adding means for storing said instructions and said logical resource driver being futher capable of fetching, in order of said instruction firing time, said instructions in each basic block, and delivering said instructions according to the logical processor number for each instruction,

a plurality of individual processor elements (PEs),

first means (650) connecting said plurality of processor elements with said logical resource driver for transferring said instructions from said logical resource driver to individually assigned processor elements, each said processor element being capable of processing said transferred instruction, a plurality of shared storage resources (660), and

second means (670) for connecting each of said processor elements with any one of said plurality of shared storage resources, each said processor elements being capable of accessing any one of said shared storage resources during said instruction processing.

13. A parallel processor system for processing natural concurrencies in streams of low level instructions contained in a program, each of said streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system comprising:

means (160) for statically adding intelligence to each instruction in each of said plurality of basic blocks,

a logical resource driver (LRD) receptive of said basic blocks corresponding to the program instruction stream from said adding means for storing said basic blocks,

a plurality of individual context free processor elements (PEs),

means (650) connecting said plurality of processor elements with said logical resource driver for transferring said instructions from said logical resource driver to individually assigned processor elements, each said processor element being capable of processing said transferred instruction,

a plurality of shared storage resources (660), and means (670) for connecting each of said processor elements with any one of said plurality of shared storage resources, each said processor elements being capable of accessing any one of said shared storage resources during said instruction processing.

14. The parallel processor system according to Claims 11, 12, or 13 in which:

said adding means is further capable of statically adding program level information to each said instruction in order to identify the different program levels contained within each said program,

said shared storage resources having a different set of resources for each said program level,

each said set of resources being identified by said statically added level information, and

said processor elements being further capable of processing each of its instructions in a set of resources identified by said statically added level information.

15. A method for parallel processing in a plurality of processor elements natural concurrencies in streams of low level instructions contained in the programs of a plurality of users, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said method comprising the steps of:

statically adding intelligence to the natural concurrencies existing within the instructions in each basic block of the programs, said step of adding for each program comprising the steps of:

(a) ascertaining the resource requirements of each instruction within each basic block to determine the natural concurrencies in each basic block,

(b) identifying logical resource dependencies between instructions,

(c) assigning condition code storages (CCs) to groups of resource dependent instructions, so that dependent instructions can execute on the same or different processor elements,

(d) determining the earliest possible instruction firing time (IFT) for each of said instructions in each of said plurality of basic blocks,

(e) adding said instruction firing times to each instruction in each of said plurality of basic blocks,

(f) assigning a logical processor number (LPN) to each instruction in each of said basic blocks,

(g) adding said logical processor numbers to each instruction in each of said basic blocks, and

(h) repeating steps (a) through (g) until all basic blocks are processed for each of said programs, and

processing the instructions having the statically added intelligence for the programs, the step of processing further comprising the steps of:

(i) delivering the instructions into logical resource drivers, each said program of a user being assigned to a different logical resource driver,

(j) selecting instructions from the logical resource drivers in a predetermined order based on the instruction firing time,

(k) storing the selected instructions into queues of the logical resource driver based on the logical processor number,

(l) generating dynamic shared context storage mapping (D-SCSM) information for each instruction,

(m) selectively connecting the queues of each logical resource driver to processor elements (PEs) said queues being connected in a predetermined order so that one instruction having the earliest instruction firing time from each queue is connected to a given processor element,

(n) processing said one instruction from each connected queue in each said connected processor element,

(o) obtaining the input data for processing said connected instruction from shared storage locations identified by said instruction in a context identified by said dynamic shared context storage mapping information,

(p) storing the results of said processing of said connected instruction in shared storage identified by said dynamic information contained in said instruction, and

(q) repeating steps (i) through (p) until all instructions are processed in each of said plurality of basic blocks for all said programs.

16. The method of Claim 15 wherein said step of statically adding intelligence further comprises the step of re-ordering said instructions in each of said basic blocks based upon said instruction firing times wherein the earliest firing times are listed first.

17. The method of Claim 15 wherein said step of statically adding intelligence further comprises the step of adding static shared context storage mapping information (S-SCSM) to each instruction to identify the program level of said instruction and wherein said step of obtaining input further comprises the step of obtaining said input data from a shared stored location identified by the aforesaid statically added information.

18. A method for parallel processing in a plurality of processor elements natural concurrencies in streams of low level instructions contained in the programs of a plurality of users, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said method comprising the steps of:

statically adding intelligence to the natural concurrencies existing within the instructions in each basic block of the programs, said step of adding for each program comprising the steps of:

(a) ascertaining the resource requirements of each instruction within each basic block to determine the natural concurrencies in each basic block,

(b) identifying logical resource dependencies between instructions,

(c) assigning condition code storages (CCs) to groups of resource dependent instructions, such that dependent instructions can execute on the same or different processor elements,

(d) determining the earliest possible instruction firing time (IFT) for each of said instructions in each of said plurality of basic blocks,

(e) adding said instruction firing times to each instruction in each of said plurality of basic blocks,

(f) assigning a logical processor number (LPN) to each instruction in each of said basic blocks,

(g) adding said logical processor numbers to each instruction in each of said basic blocks, and

(h) repeating steps (a) through (h) until all basic blocks are processed for each of said programs, and processing the instructions having the statically added intelligence for said programs on a plurality of processing elements (PEs).

19. The method of Claim 18 wherein said step of statically adding intelligence further comprises the step of re-ordering said instructions in each of said basic blocks based upon said instruction firing times wherein the earliest firing times are listed first.

20. The method of Claim 15 wherein said step of statically adding intelligence further comprises the step of adding static shared context storage mapping (S-SCSM) information to each instruction to identify the program level of said instruction.

21. A method for parallel processing natural concurrencies in streams of low level instructions contained in the programs of a plurality of users located in a system having a plurality of processor elements and shared storage locations, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said method comprising the steps of:

statically adding intelligence to the instructions in each basic block of the programs, said added intelligence identifying the natural concurrencies within each basic block, said added intelligence having at least an instruction firing time (IFT) and a logical processor number (LPN), and processing the instructions having the statically added intelligence for the programs, the step of processing further comprising the steps of:

(a) delivering the instructions into the system, each said user being assigned to a different context in said system,

(b) dynamically generating shared context state mapping (D-SCSM) information for each instruction identifying said context,

(c) separately storing the delivered instructions in the system based on the logical processor number (LPN),

(d) selectively connecting the separately stored instructions to the processor elements assigned to the logical processor number for the instructions, said separately stored instructions being connected in a predetermined order so that one instruction having the earliest instruction firing time from each of the separately stored instructions is connected to a given processor element,

(e) processing said one instruction from each connected separately stored instructions in each said connected processor element,

(f) obtaining the input data for processing said connected instruction from a shared storage location identified by said shared context state mapping information,

(g) storing the results of said processing of said connected instruction in the identified shared storage located, and

(h) repeating steps (a) through (g) until all instructions are processed in each of said plurality of basic blocks for all of said program.

22. A method for parallel processing in a system natural concurrencies in streams of low level instructions contained in a program, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system having a plurality of processor elements and a plurality of shared storage locations, said method comprising the steps of:

statically adding intelligence to the natural concurrencies existing within the instructions in each basic block, said step of adding comprising the steps of:

(a) ascertaining the resource requirements of each instruction within each basic block to determine the natural concurrencies in each basic block,

(b) identifying logical resource dependencies between instructions,

(c) assigning condition code storages (CCs) to groups of resource dependent instructions, such that dependent instructions can execute on the same or different processor elements,

(d) determining the earliest possible instruction firing time (IFT) for each of said instructions in each of said plurality of basic blocks,

(e) adding said instruction firing times to each instruction in each of said plurality of basic blocks,

(f) assigning a logical processor number (LPN) to each instruction in each of said basic blocks,

(g) adding said logical processor numbers to each instruction in each of said basic blocks, and

(h) repeating steps (a) through (g) until all basic blocks are processed for said program, and

processing the instructions having the statically added intelligence in the system, the step of processing further comprising the steps of:

(i) separately storing the delivered instructions in the system based on the logical processor number,

(j) selectively connecting the separately stored instructions to the processor elements (PEs) assigned to the logical processor number, said separately stored instructions being connected in a predetermined order so that one instruction having the earliest instruction firing time is connected to a given processor element,

(k) processing said one instruction from each of the separately stored instructions in each said connected processor element, and

(l) repeating steps (i) through (k) until all instructions are processed in each of said plurality of basic blocks for said program.

23. A method for parallel processing in a system natural concurrencies in streams of low level instructions contained in a program, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system having a plurality of processor elements, said method comprising the steps of:

statically adding intelligence to the natural concurrencies existing within the instructions in each basic block, said step of adding comprising the steps of:

(a) ascertaining the resource requirements of each instruction within each basic block to determine the natural concurrencies in each basic block,

(b) identifying logical resource dependencies between instructions,

(c) assigning condition code storages (CCs) to groups of resource dependent instructions, such that dependent instructions can execute on the same or different processor elements,

(d) determining the earliest possible instruction firing time (IFT) for each of said instructions in each of said plurality of basic blocks,

(e) adding said instruction firing times (IFTs) to each instruction in each of said plurality of basic blocks,

(f) assigning a logical processor number (LPN) to each instruction in each of said basic blocks,

(g) adding said logical processor numbers to each instruction in each of said basic blocks, and

(h) repeating steps (a) through (h) until all basic blocks are processed for said program, and

processing the statically added instructions having the added intelligence with a plurality of all processor elements (PEs).

24. The method according to Claims 22 or 23 wherein said step of statically adding intelligence further comprises the step of re-ordering said instructions in each of said basic blocks based upon said instruction firing times wherein the earliest firing times are listed first.

25. The method according to Claims 22 or 23 wherein said step or statically adding intelligence further comprises the step of adding program level information to identify the program level of said instruction.

26. A method for parallel processing in a system natural concurrencies in streams of low level instructions contained in a program, each of the streams having a plurality of single entry-single exit (SESE) basic blocks (BBs), said system having a plurality of processor elements and a plurality of shared storage locations, said method comprising the steps of:

statically adding intelligence to the natural concurrencies existing within the instructions in each basic block, said step of adding comprising the steps of:

(a) determining the earliest possible instruction firing time (IFT) for each of said instructions in each of said plurality of basic blocks,

(b) adding said instruction firing times to each instruction in each of said plurality of basic blocks,

(c) assigning a logical processor number (LPN) to each instruction in each of said basic blocks, and

(d) adding said logical processor numbers to each instruction in each of said basic blocks, and

processing the instructions having the statically added intelligence, the step of processing further comprising the steps of:

(e) separately storing the delivered instructions in the system based on the logical processor number,

(f) selectively connecting said separately stored instructions to processor elements (PEs) assigned to a given logical processor number, said instructions connected in a predetermined order so that one instruction having the earliest instruction firing time from each of said separately stored instructions is connected to a given processor element,

(g) processing said one connected instruction in each said connected processor element,

(h) obtaining the input data for processing said connected instruction from a shared storage location identified by said instruction,

(i) storing the results of said processing of said connected instruction in a shared storage located in said identified context for said instruction, and

(j) repeating steps (e) through (i) until all instructions are processed in each of said plurality of basic blocks for said program.

27. A method for parallel processing natural concurrencies in a program with a plurality of processing elements (PEs), said program having a plurality of single entry-single exit (SESE) basic blocks (BBs) with each of said basic blocks (BBs) having a stream of instructions, said method comprising the steps of:

determining the natural concurrencies within said instruction stream in each of said basic blocks (BBs) in said program,

adding intelligence to each instruction in each said basic block in response to the determination of said natural concurrencies, said added intelligence at least comprising an instruction firing time (IFT) and a logical processor number (LPN), and processing the instructions having said added intelligence in said plurality of processing elements, each of said plurality of processing elements receiving all instructions in the order of the instruction having earliest instruction firing time first.

28. The method of Claim 27 wherein the step of adding intelligence further comprises the adding of static shared context storage mapping (S-SCSM) information and wherein the step of processing further comprises the step of processing each instruction in communication with shared resources as identified by each said instruction's static shared context storage mapping information, so each program routine can access resources at other procedural levels in addition to the resources at that routine's procedural level.

29. The method of Claim 27 wherein the step of determining the natural concurrencies within each basic block comprises the steps of:

ascertaining the resource requirements of each instruction within each of said basic blocks,

(a) identifying logical resource dependencies between instructions, and

(b) assigning condition code storages (CCs) to groups of resource dependent instructions,

30. The method of Claim 27 wherein the step of adding intelligence to each instruction further comprises the steps of:

determining the earliest possible instruction firing time for each of said instructions in each of said plurality of basic blocks,

adding said instruction firing times (IFTs) to each instruction in each of said plurality of basic blocks in response to said determination, and

reordering said instructions in each of said basic blocks based upon said instruction firing times.

31. The method of Claim 27 wherein the step of adding intelligence to each instruction further comprises the steps of:

determining the earliest possible instruction firing time for each of said instructions in each of said plurality of basic blocks, and

adding said instruction firing times (IFTs) to each instruction in each of said plurality of basic blocks in response to said determination.

32. The method according to Claims 30 or 31 further comprising the steps of:

assigning a logical processor number to each instruction in each of said basic blocks, and

adding said assigned logical processor number to each instruction in each of said basic blocks in response to said assignment.

33. The method of Claim 27 further comprising the step of forming execution sets (ESs) of basic blocks in response to said step of adding said intelligence wherein branches from any given basic block within a given execution set to a basic block in another execution set is statistically minimized.

34. The method of Claim 27 wherein the step of processing further comprises the steps of:

separately storing the instructions with said added intelligence based on the logical processor number, each of said separately stored instructions containing instructions having only the same logical processor number,

selectively connecting said separately stored instructions to said processing elements, and

each said processing element receiving each instruction with the earliest instruction firing time first.

35. The method of Claim 34 wherein the step of processing said instruction received by an individual processing element further comprises the steps of:

obtaining the input data for processing said received instruction from shared storage locations identified by said instruction, storing the results from processing said received instruction in a shared storage identified by said instruction, and

repeating the aforesaid steps for the next instruction until all instructions are processed.

36. A method for parallel processing natural concurrencies in a program with a plurality of processing elements (PEs), said processing elements having access to input data located in a plurality of shared resource locations, said program having a plurality of single entry-single exit (SESE) basic blocks (BBs) with each of said basic blocks (BBs) having a stream of instructions, said method comprising the steps of:

ascertaining the resource requirements of each instruction within each of said basic blocks,

(a) identifying logical resource dependencies between instructions,

(b) assigning condition code storages (CCs) to groups of resource dependent instructions, such that dependent instructions can execute on the same or different processor elements,

determining the earliest possible instruction firing time (IFT) for each of the instructions in said plurality of basic blocks,

adding said instruction firing times (IFTs) to each instruction in each of said plurality of basic blocks in response to said determination,

assigning a logical processor number (LPN) to each instruction in each of said basic blocks in response to said determination,

adding said assigned logical processor number (LPN) to each instruction in each of said plurality of basic blocks in response to said assignment,

separately storing the instructions with said added instruction firing time and said added logical processor numbers based on the logical processor number, each of said separately stored instructions containing instructions having only the same logical processor number, and

selectively connecting said separately stored instructions to said processing elements based on the logical processor number, and

each said processing element receiving each instruction in said connected queue having the earliest instruction firing time first, said processing element being capable of performing the steps of:

(a) obtaining said input data for processing said received instruction from a shared storage location in said plurality of shared resource locations identified by said instruction,

(b) storing the results based upon the aforesaid step of processing in a shared storage location in said plurality of shared resource locations identified by said received instruction, and

(c) repeating the aforesaid steps (a) and (b) for the next received instruction until all instructions are processed.

37. The method of Claim 36 further comprising the steps of forming execution sets of basic blocks in response to said steps of adding said instruction firing times and logical processor numbers wherein branches from any given basic block within a given execution set to a basic block in another execution set is statistically minimized.

38. The method of Claim 36 further comprising the steps of:

adding shared context storage mapping (S-SCSM) information to each instruction, and

wherein said step of processing comprises processing each instruction in communication with a set of shared resources as identified by said shared context storage mapping information so each program routine can access its set of resources in addition to the routine's procedural level resources.

39. A method for parallel processing natural concurrencies in a program with a plurality of processing elements (PEs), said processing elements having access to input data located in a plurality of shared resource locations, said program having a plurality of single entry-single exit (SESE) basic blocks (BBs) with each of said basic blocks (BBs) having a stream of

instructions, said method comprising the steps of:

ascertaining the resource requirements of each instruction within each of said basic blocks,

(a) identifying logical resource dependencies between instructions,

(b) assigning condition code storages (CCs) to groups of resource dependent instructions, such that dependent instructions can execute on the same or different processor elements

determining the earliest possible instruction firing time (IFT) for each of the instructions in each of said plurality of basic blocks,

adding said instruction firing times to each instruction in each of said plurality of basic blocks in response to said determination,

assigning a logical processor number (LPN) to each instruction in each of said basic blocks in response to said determination,

adding said assigned logical processor number to each instruction in each of said plurality of basic blocks in response to said assignment,

forming execution sets (ESs) of basic blocks in response to said steps of adding said instruction firing times and logical processor numbers,

(i) separately storing the instructions contained within a given formed execution set based on the logical processor number, each of said separately stored instructions containing instructions having only the same logical processor number,

(ii) selectively connecting said separately stored instructions to said processing elements based on the logical processor number,

(iii) each said processing element receiving each instruction having the earliest instruction firing time (IFT) first, said processing element being capable of performing the steps of:

(a) obtaining said input data for processing said received instruction from a shared storage location in said plurality of shared resource locations identified by said instruction,

(b) storing the results based upon the aforesaid step of processing in a shared storage location in said plurality of shared resource locations identified by said received instruction,

(c) repeating the aforesaid steps (a) and (b) for the next received instruction until all instructions are processed, and

(iv) repeating the aforesaid steps (i) through (iii) for all execution sets that are processed.

40. A method for parallel processing in a system natural concurrencies in a plurality programs of different users with a plurality of processing elements (PEs), each of said programs

having a plurality of single entry-single exit (SESE) basic blocks (BBs) with each of said basic blocks (BBs) having a stream of instructions, said method comprising the steps of:

determining the natural concurrencies within said instruction stream in each of said basic blocks in each said program,

adding intelligence to each instruction in each said basic block in response to the determination of said natural concurrencies, said added intelligence at least comprising an instruction firing time (IFT) and a logical processor number (LPN),

processing the instructions having said added intelligence from said programs in said plurality of processing elements, each of said plurality of processing elements receiving all instructions in the order of the instruction having earliest instruction firing time first for each said program, each said processing element being capable of processing one instruction from each program in a predetermined order.

41. The method of Claim 40 in which the step of processing further comprising the steps of: dynamically adding context information (D-SCSM) to each instruction, said dynamically added information identifying the context in said system assigned to each said program, each of said processing elements being further capable of processing each of its instructions only in the context identified by said added context information.

42. The method of Claim 40 wherein the step of adding intelligence further comprises the adding of program level information to each instruction involved in program level transfers and wherein the step of processing further comprises processing each instruction in communication with a set of shared registers as identified by said instruction's program level information, so each program routine can access its set of registers in addition to the routine's procedural level resources.

43. The method of Claim 40 wherein the step of determining the natural concurrencies within each basic block of each program comprises the steps of:

ascertaining the resource requirements of each instruction within each of said basic blocks, and

(a) identifying logical resource dependencies between instructions,

(b) assigning condition code storages (CCs) to groups of resource dependent instructions, so that dependent instructions can execute on the same or different processor elements,

44. The method of Claim 40 wherein the step of adding intelligence to each instruction in each said program further comprises the steps of:

determining the earliest possible instruction firing time for each of said instructions in each of said plurality of basic blocks, adding said instruction firing times to each instruction in each of said plurality of basic blocks in response to said determination, and

reordering said instructions in each of said basic blocks based upon said instruction firing times.

45. The method of Claim 40 wherein the step of adding intelligence to each instruction in each said program further comprises the steps of:

determining the earliest possible instruction firing time for each of said instructions in each of said plurality of basic blocks, and

adding said instruction firing times to each instruction in each of said plurality of basic blocks in response to said determination.

46. The method according to Claims 44 or 45 further comprising the steps of:

assigning a logical processor number to each instruction in each of said basic blocks, and

adding said assigned logical processor number to each instruction in each of said basic blocks in response to said assignment.

47. The method of Claim 40 further comprising the step of forming execution sets (ESs) of basic blocks for each said program

in response to said step of adding said intelligence wherein branches from any given basic block within a given execution set to a basic block in another execution set is statistically minimized.

48. The method of Claim 40 wherein the step of processing further comprises the steps of:

separately storing the instructions with said added intelligence into a plurality of sets wherein at least one set contains at least one program and wherein said separate storage in each set is based on the logical processor number,

selectively connecting said separately stored instructions for one of said sets to said processing elements based on said logical processor number, and

each said processing element receiving each instruction having the earliest instruction firing time first.

49. The method of Claim 48 wherein the step of receiving said instruction from said separately stored instructions in said sets by an individual processing element further comprises the steps of:

obtaining the input data for processing said received instruction from shared storage locations identified by said instruction and by said set,

storing the results based from processing said received instruction in a shared storage identified by said received instruction and said set, and

repeating the aforesaid steps for the next received instruction from said next set based upon said predetermined order until all instructions are processed in said sets.

50. A method for parallel processing natural concurrencies in a plurality of programs with a plurality of processing elements (PEs), said processing elements having access to input data located in a plurality of shared resource locations, each of said programs having a plurality of single entry-single exit (SESE) basic blocks (BBs) with each of said basic blocks (BBs) having a stream of instructions, said method comprising the steps of:

ascertaining the resource requirements of each instruction within each of said basic blocks for each said program,

determining the earliest possible instruction firing time (IFT) for each of the instructions in each of said plurality of basic blocks,

adding said instruction firing times to each instruction in each of said plurality of basic blocks in response to said determination,

assigning a logical processor number (LPN) to each instruction in each of said basic blocks in response to said determination,

adding said assigned logical processor number to each instruction in each of said plurality of basic blocks in response to said assignment,

separately storing the instructions with said added instruction firing time and said added logical processor numbers into a plurality of sets wherein at least one set contains at least one of said programs and wherein said separate storage in each set is based on the logical processor number,

selectively connecting said separately stored instructions for one of said sets to said processing elements based on the logical processor number, and

each said processing element receiving each instruction in each said connected set with the earliest instruction firing time (IFT) first, said processing element being capable of performing the steps of:

(a) obtaining said input data for processing said received instruction from a shared storage location in said plurality of shared resource locations identified by said instruction and by said set,

(b) storing the results based from processing said received instruction in a shared storage location in said plurality of shared resource locations identified by said received instruction and by said set, and

(c) repeating the aforesaid steps (a) and (b) for the next received instruction from said next set based upon a predetermined order until all instructions are processed.

51. The method of Claim 50 further comprising the steps of

forming execution sets (ESs) of basic blocks in response to said steps of adding said instruction firing times and logical processor numbers wherein branches from any given basic block within a given execution set to a basic block in another execution set is statistically minimized.

52. The method of Claim 50 wherein the step of adding intelligence further comprises the adding of the static shared context storage mapping (S-SCSM) information and wherein the step of processing further comprises the step of processing each instruction in communication with a set of shared registers as identified by said instruction's static shared context state mapping information, so each program routine can access its set of registers in addition to the procedural level resource.

53. A system for parallel processing natural concurrencies in a program, said program having a plurality of single entry-single exist (SESE) basic blocks (BBs) wherein each of said basic blocks (BBs) contains a stream of instructions, said system comprising:

means (160) receptive of said plurality of basic blocks for determining said natural concurrencies within said instruction stream for each of said basic blocks, said determining means being further capable of adding at least an instruction firing time (IFT) and a logical processor number (LPN) to each instruction in said determined natural concurrencies so that all processing resources required by any given instruction are allocated in advance of processing,

means (620) receptive of said basic blocks having said added instruction firing times and logical processor numbers for separately storing said received instructions based upon said logical processor number, and

a plurality of processing elements (PEs), connected to said storing means based upon said logical processor numbers, each of said processing elements being capable of processing its received instructions in the order of processing those instructions with the earliest instruction firing time first.

54. The parallel processor system of ~~Claim~~ 53 in which:

said determining means is ~~further~~ capable of program level information (S-SCSM) information to each said instruction, said information containing level information for each said instruction in order to identify the different program levels contained within said program,

a plurality of registers, said registers having a different set of registers for each said program level, each said set of registers being identified by said information, and

said processor elements being further capable of processing each of its instructions in a set of registers identified by said information.

55. The system of Claim 53 wherein said determining means is

further capable of forming basic blocks containing said added instruction firing times and logical processor numbers into execution sets (ESs) wherein branches from any given basic block within a given execution set out of said given execution set to a basic block in another execution set is statistically minimized.

56. The system of Claim 55 wherein said determining means is further capable of attaching header information to each formed execution set, said header at least comprising:

(a) the address of the start of said instructions, and

(b) the length of the execution set.

57. The system of Claim 55 wherein said storing means further comprises:

a plurality of caches (1522) receptive of said execution sets for storing said instructions,

means (1544, 1560, 1570) connected to said caches for delivering said instructions stored in each said caches (1522) to said plurality of processing elements (PEs), and

means (1512, 1516, 1548) connected to said caches and said delivering means for controlling said storing and said delivery of instructions, said controlling means being further capable of executing the branches from individual basic blocks.

58. The system of Claim 53 wherein each of said plurality of processing elements is context free in processing a given

instruction.

59. The system of Claim 58 wherein said plurality of shared resources comprises:

a plurality of register files, and

a plurality of condition code files, said plurality of register files and said plurality of condition code files together with said storing means (620) receptive of said basic blocks being capable of storing all necessary context information for any given instruction during the processing of said instruction.

60. A system for parallel processing natural concurrencies in a program, said program having a plurality of single entry-single exit (SESE) basic blocks (BBs) wherein each of said basic blocks contains a stream of instructions, said system comprising:

means (160) receptive of said plurality of basic blocks for determining said natural concurrencies within said instruction stream for each of said basic blocks, said determining means being further capable of adding timing and processor information to each instruction in said determined natural concurrencies so that all processing resources required by any given instruction are allocated in advance of processing,

means (620) receiving said basic blocks (BBs) having said added timing and processor information for storing said received instructions,

a plurality of processor elements (PEs), means (650) for selectively connecting said plurality of processor elements to said storing means,

a plurality of shared resources (660),

means (670) for selectively interconnecting said plurality of processor elements (PEs) with said plurality of shared resources (660), said storing means being capable of delivering instructions in order of the earliest time first, based upon said timing information, to the processor elements over said connecting means and into said processor elements, and

said processor elements being capable of processing each received instruction from said storing means (620), said processor elements being connected to said shared resources identified in each said instruction so that all resource and context information pertaining to said instruction is stored in said plurality of shared resources and said storing means (620).

61. The parallel processor system of Claim 60 in which:

said determining means is further capable of program level information (S-SCSM) to each said instruction, said information containing level information for each said instruction in order to identify the different program levels contained within said program,

said shared resources having a different set of registers for each program level, each said set of registers being identified by said information, and

said processor elements being further capable of processing each of its instructions in a set of registers identified by said information.

62. The system of Claim 60 wherein said determining means is further capable of forming basic blocks containing said added timing and processor information into execution sets (ESs) wherein branches from any given basic block within a given execution set out of said given execution set to a basic block in another execution set is statistically minimized.

63. The system of Claim 62 wherein said determining means is further capable of attaching header information to each formed execution set, said header at least comprising:

(a) the address of the start of said instructions, and

(b) the length of the execution set.

64. The system of Claim 60 wherein said storing means further comprises:

a plurality of caches (1522) receptive of said execution sets for storing said instructions,

means (1544, 1560, 1570) connected to said caches for delivering said instructions stored in each said caches to said plurality of processor elements, and

means (1512, 1516, 1548) connected to said caches and said delivering means for controlling said storing and said delivery of instructions, said controlling means being further capable of executing the branches from individual basic blocks.

65. The system of Claim 60 wherein said determining means is further capable of adding level information to instructions pertaining to the different program levels contained within said program, and wherein said processor elements are further capable of processing each of its instructions in a set of shared resources identified by each said instructions level information.

66. The system of Claim 60 wherein each of said plurality of processor elements is context free in processing a given instruction.

67. The system of Claim 66 wherein said plurality of shared resources comprises:

a plurality of register files, and

a plurality of condition code files, said plurality of register files and said plurality of condition code files together with said storing means (620) receptive of said basic block being capable of storing all necessary context for processing any given instruction.

68. A system for parallel processing natural concurrencies in a program, said program having a plurality of single entry-single exit (SESE) basic blocks (BBs) wherein each of said basic blocks (BBs) contains a stream of instructions, said system

~~comprising:~~

~~means (160) receptive of said plurality of basic blocks for determining said natural concurrencies within said instruction stream for each of said basic blocks, said determining means being further capable of adding timing, processor, and resource access information to each instruction in said determined natural concurrencies so that all processing resources required by any given instruction are allocated in advance of processing, said determining means being further capable of forming basic blocks containing said added instruction firing times and logical processor numbers into execution sets (ESs) wherein branches from any given basic block within a given execution set out of said given execution set to a basic block in another execution set is statistically minimized,~~

~~means (620) receptive of said execution sets having said added information for storing said instructions,~~

~~a plurality of context-free processor elements connected to said storing means, and~~

~~a plurality of shared resources connected to said plurality of context-free processor elements, said processor elements being capable of processing its instructions based upon said processor, timing, and resource access information from said storing means, said processing elements being capable of obtaining all necessary context data from said plurality of shared resources based upon locations set forth in said instructions for processing said instruction and being capable of delivering all necessary~~

context data to said plurality of shared resource locations based upon locations identified in said instructions.

69. A system for accessing data between levels in a program, said system comprising:

means (160) receptive of said program for determining the levels contained therein, said determining means being further capable of adding information (S-SCSM) to all instructions involved in said data accesses in said program, said information at least identifying the calling and called levels,

a predetermined number of shared resources (660), each of said shared resources being independent of each other, and

means (620, 640) operative on the instructions in said program for processing said program, said processing means being capable of fully processing all of the instructions based upon said information within a procedural level of said program in the said plurality of shared resources corresponding to said procedural level, said processing means upon completion of processing the aforesaid procedural level of said program being further capable of storing the results in at least one of said shared resources corresponding to a different procedural level of said program based upon said information.

70. A system for accessing data between levels in a plurality of programs utilized by a number of users, said system comprising:

means (160) receptive of each of said programs for determining the levels contained therein, said determining means being further capable of adding first information to all instructions involved in said data accesses within each of said programs, said first information at least identifying the calling level and the called level,

means (620) receptive of each of said programs containing said first added information from said determining means for adding second information to each instruction of said programs identifying at least the user context,

a plurality of contexts (660) one for each user, each of said contexts having a predetermined number of shared resources, each of said shared resources being independent of each other, and

means (600, 640) operative on the instructions in each of said programs for processing the aforesaid instructions from each of said programs in a predetermined order, said processing means being capable of fully processing each of the aforesaid instructions based upon said first added information within a procedural level for each said program in the plurality shared resources in the shared resources corresponding to said called procedural level and located in a context identified by said second added information, said processing means upon completion of processing the aforesaid procedural level of the aforesaid program being further capable of storing the results in at least one resource in the shared resource corresponding to a different procedural level of the aforesaid program based upon said first added information.

Awc1 71. A system for executing branches in single entry-single exist (SESE) basic blocks (BBs) contained within a program, said system comprising:

means (620) receptive of said program for determining the branch instruction within each said basic block of said program, said determining means being further capable of adding information (IFT) to said branch instruction,

means (620, 640) operative on the instructions in each said basic block for processing said instructions, and

means (620, 1548) operative on said branch instruction in said basic block for completing the execution of said branch instruction during the same time (IFT) as said processing means is processing the last instruction in said basic block so that the execution of said branch instruction occurs in parallel with the execution of said instructions in said basic block in order to speed up the overall processing of said program by said system.

72. A system for executing branches in single entry-single exit (SESE) basic blocks (BBs) in a plurality of programs utilized by a number of users, said system comprising:

means (160) receptive of each said programs for determining the branch instruction within each said basic block of each of said programs, said determining means being further capable of adding information (IFT) to said branch instructions,

~~means (620, 640) operative on the instructions in each said basic block of each said program for processing said programs, and~~

~~means (620, 1548) operative on said branch instructions in each said basic block for completing the execution of said branch instruction during the same time (IFT) as said processing means is processing the last instruction in said basic block for a given program so that the execution of said branch instruction occurs in parallel with the execution of said instructions in said basic block in order to speed up the overall processing all of said programs by said system.~~

~~73. A multiprocessor system for processing a plurality of programs of different users, said system comprising:~~

~~a plurality of logical resource drivers (LRDs) each of said logical resource drivers being operative on one of said programs for dynamically assigning information to each said instruction in said one program, said information identifying at least the user context of said one program,~~

~~a plurality of sets of shared resources (660), each of said sets being assigned to a given context, and~~

~~a plurality of processor elements (PEs) connected to said plurality of sets of shared resources and receptive of instructions in a predetermined order from said plurality of logical resource drivers for processing said programs, each of said processor elements being context free and each of said~~

~~processor elements being selectively interconnected with the set of shared resources identified by said user context information attached to each said instruction and information contained within said instructions in order to receive all data necessary for the processing of said instruction.~~

add A^2

add B'